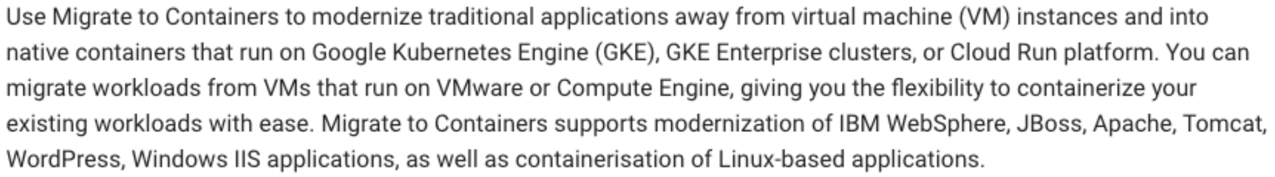
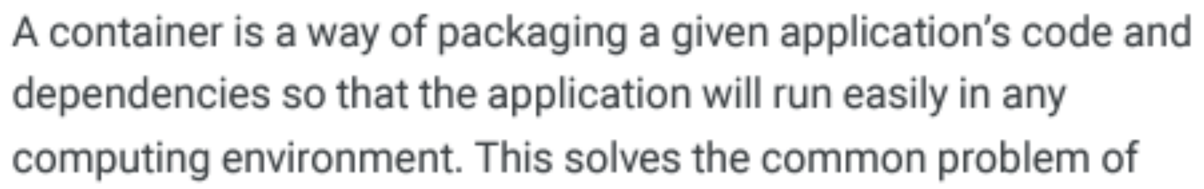
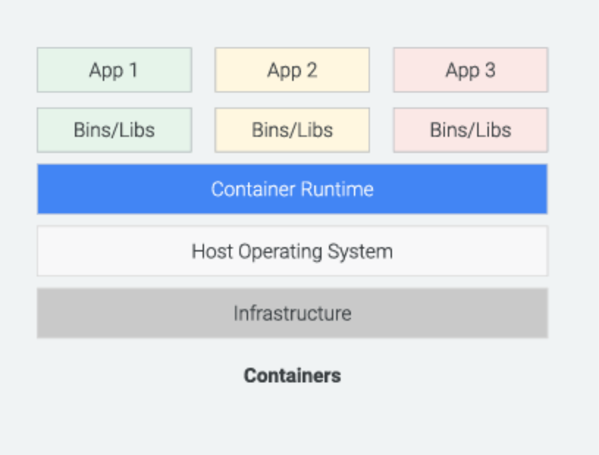
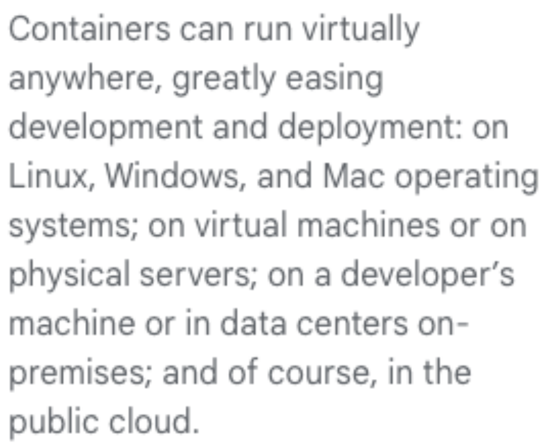
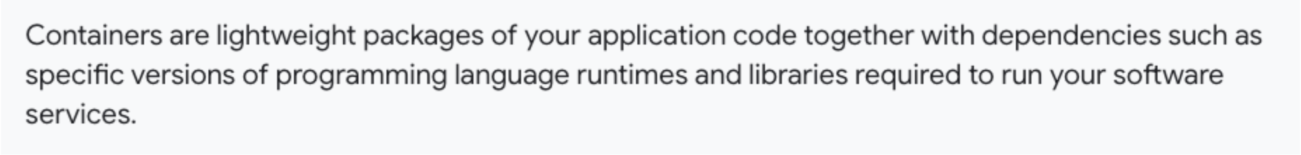
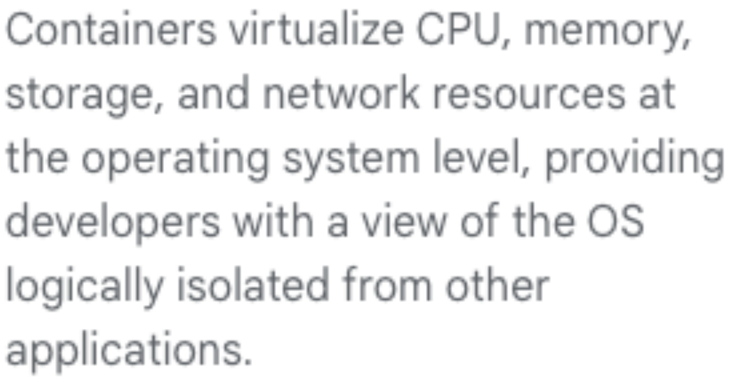
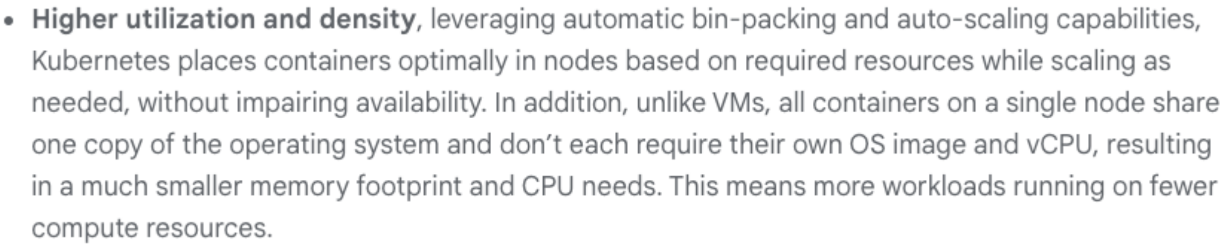
U.S. Patent No. 7,784,058 ("'058 Patent")

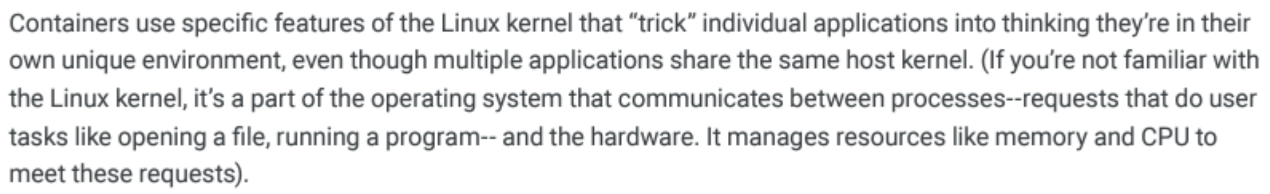
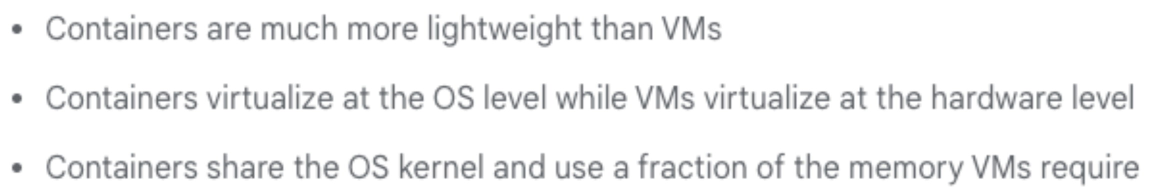
Accused Instrumentalities: Google's "Migrate to Containers," and all versions and variations thereof since the issuance of the asserted patent.

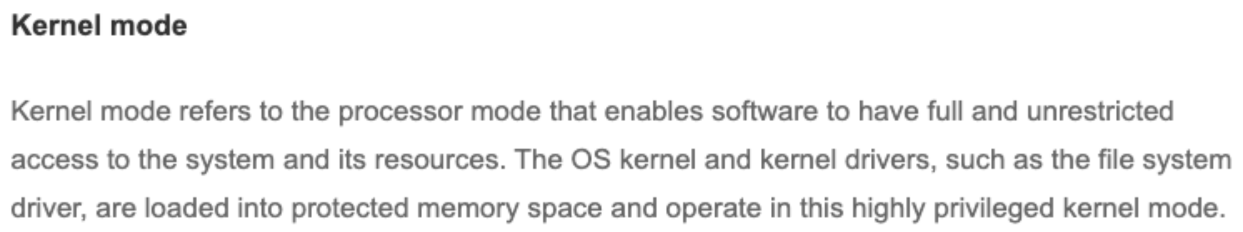
Claim 1

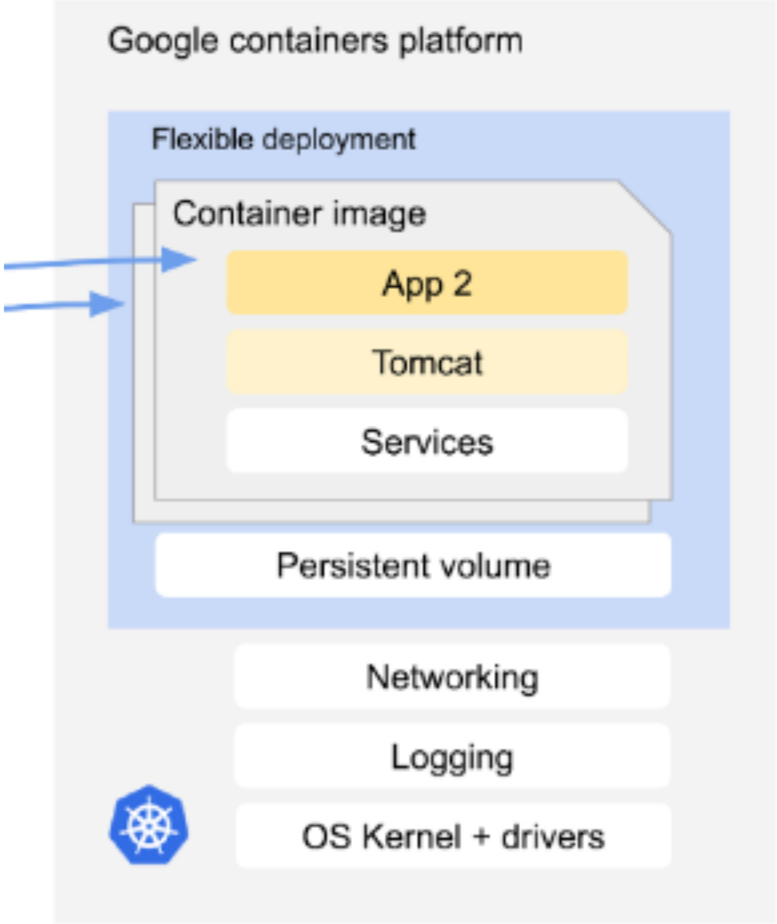
Claim 1	Accused Instrumentalities
[1pre] 1. A computing system for executing a plurality of software applications comprising:	To the extent the preamble is limiting, each Accused Instrumentality comprises or constitutes a computing system for executing a plurality of software applications as claimed. *See* claim limitations below. *See also, e.g.*:  Use Migrate to Containers to modernize traditional applications away from virtual machine (VM) instances and into native containers that run on Google Kubernetes Engine (GKE), GKE Enterprise clusters, or Cloud Run platform. You can migrate workloads from VMs that run on VMware or Compute Engine, giving you the flexibility to containerize your existing workloads with ease. Migrate to Containers supports modernization of IBM WebSphere, JBoss, Apache, Tomcat, WordPress, Windows IIS applications, as well as containerisation of Linux-based applications. https://cloud.google.com/migrate/containers/docs/getting-started.  A container is a way of packaging a given application's code and dependencies so that the application will run easily in any computing environment. This solves the common problem of

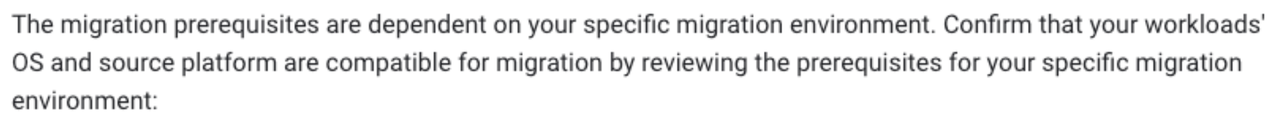
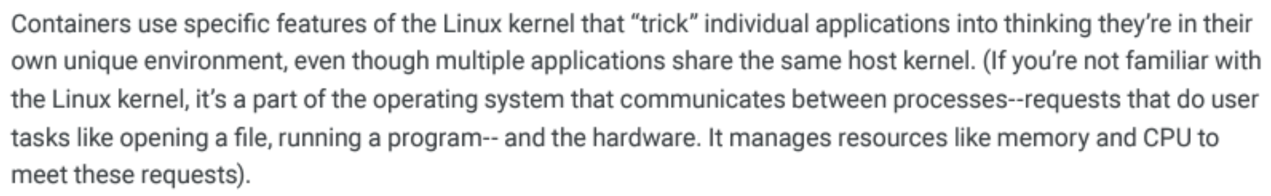
Claim 1	Accused Instrumentalities
	 App 1 App 2 App 3 Bins/Libs Bins/Libs Bins/Libs Container Runtime Host Operating System Infrastructure Containers https://services.google.com/fh/files/misc/why_container_security_matters.pdf  Containers can run virtually anywhere, greatly easing development and deployment: on Linux, Windows, and Mac operating systems; on virtual machines or on physical servers; on a developer's machine or in data centers on-premises; and of course, in the public cloud.

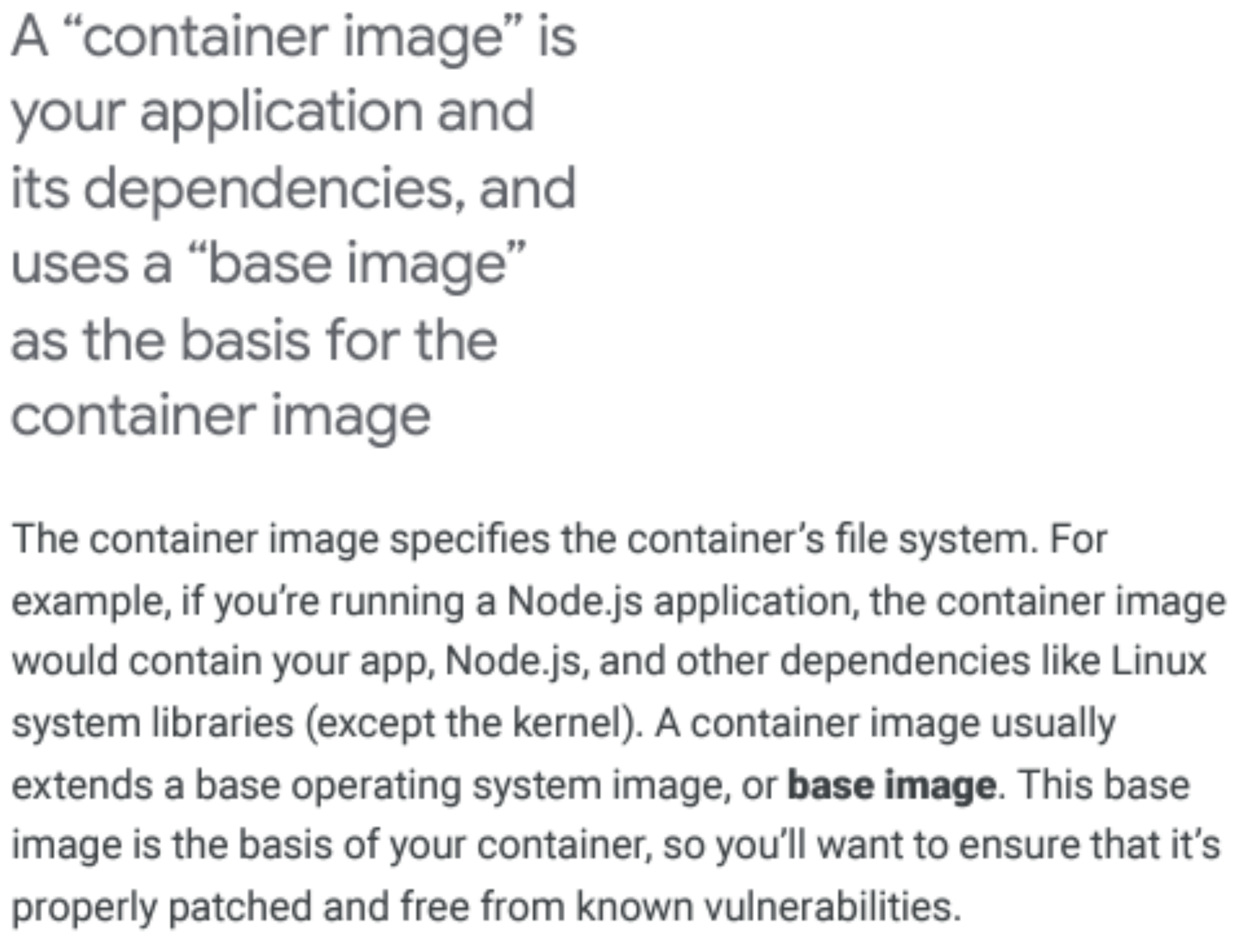
Claim 1	Accused Instrumentalities
	 Containers are lightweight packages of your application code together with dependencies such as specific versions of programming language runtimes and libraries required to run your software services. https://cloud.google.com/learn/what-are-containers
[1a] a) a processor;	Each Accused Instrumentality comprises a processor. *See, e.g.*:  Containers virtualize CPU, memory, storage, and network resources at the operating system level, providing developers with a view of the OS logically isolated from other applications. https://cloud.google.com/learn/what-are-containers  • **Higher utilization and density**, leveraging automatic bin-packing and auto-scaling capabilities, Kubernetes places containers optimally in nodes based on required resources while scaling as needed, without impairing availability. In addition, unlike VMs, all containers on a single node share one copy of the operating system and don't each require their own OS image and vCPU, resulting in a much smaller memory footprint and CPU needs. This means more workloads running on fewer compute resources.

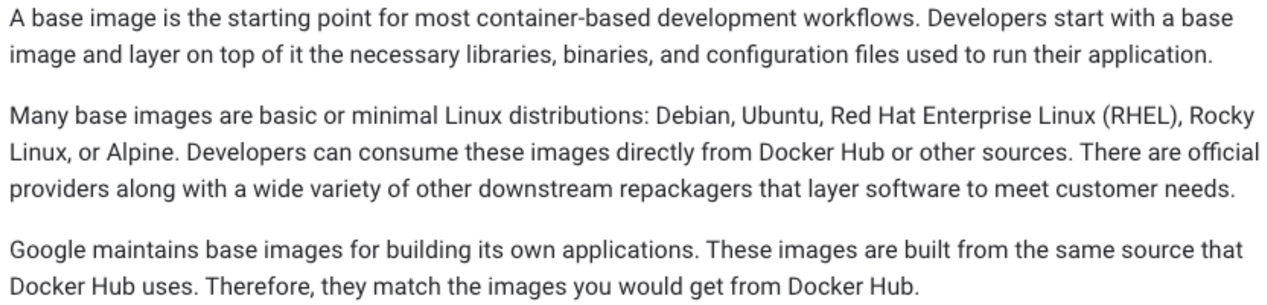
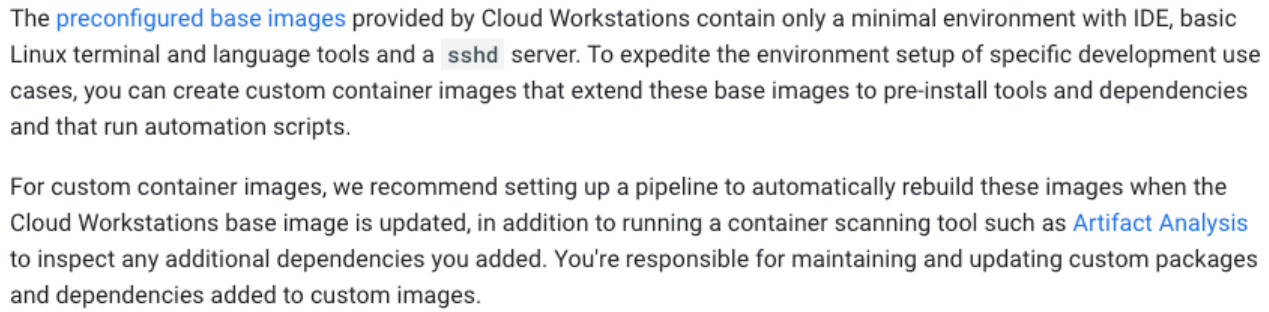
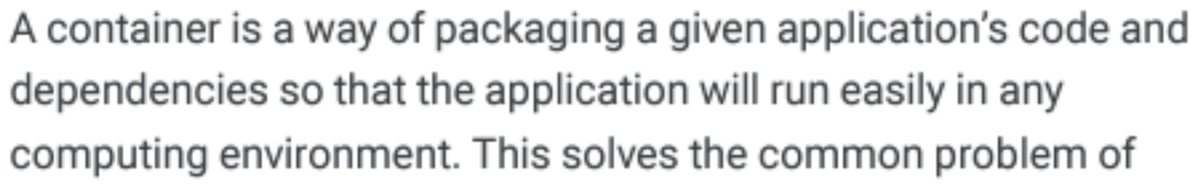
Claim 1	Accused Instrumentalities
	https://cloud.google.com/blog/products/containers-kubernetes/how-migrate-for-anthos-improves-vm-to-container-migration  Containers use specific features of the Linux kernel that "trick" individual applications into thinking they're in their own unique environment, even though multiple applications share the same host kernel. (If you're not familiar with the Linux kernel, it's a part of the operating system that communicates between processes--requests that do user tasks like opening a file, running a program-- and the hardware. It manages resources like memory and CPU to meet these requests). https://services.google.com/fh/files/misc/why_container_security_matters.pdf
[1b] b) an operating system having an operating system kernel having OS critical system elements (OSCSEs) for running in kernel mode using said processor; and,	Each Accused Instrumentality comprises an operating system having an operating system kernel having OS critical system elements (OSCSEs) for running in kernel mode using said processor. *See, e.g.*:  • Containers are much more lightweight than VMs • Containers virtualize at the OS level while VMs virtualize at the hardware level • Containers share the OS kernel and use a fraction of the memory VMs require https://cloud.google.com/learn/what-are-containers

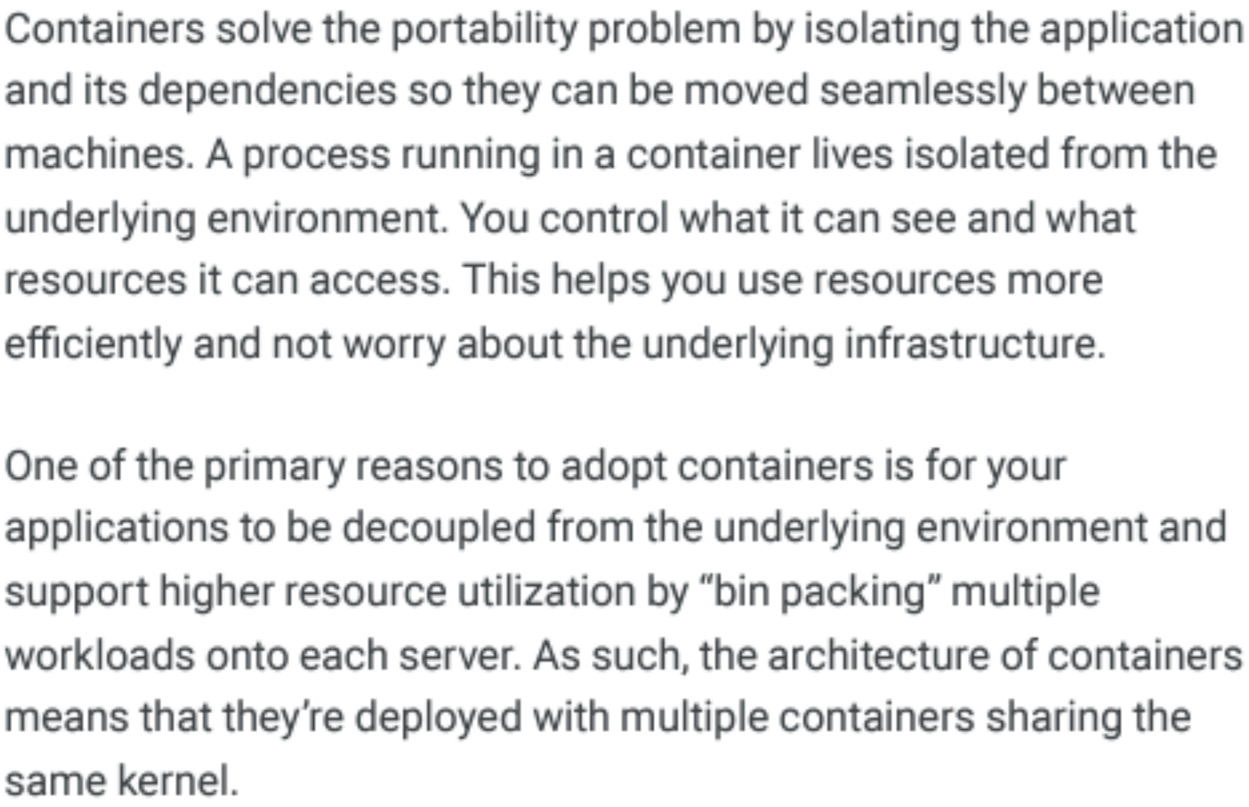
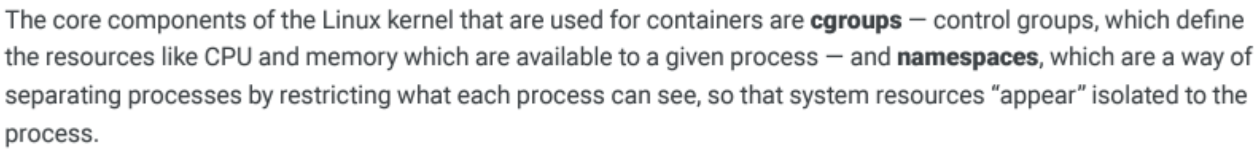
Claim 1	Accused Instrumentalities
	 **Kernel mode** Kernel mode refers to the processor mode that enables software to have full and unrestricted access to the system and its resources. The OS kernel and kernel drivers, such as the file system driver, are loaded into protected memory space and operate in this highly privileged kernel mode. https://www.techtarget.com/searchdatacenter/definition/kernel

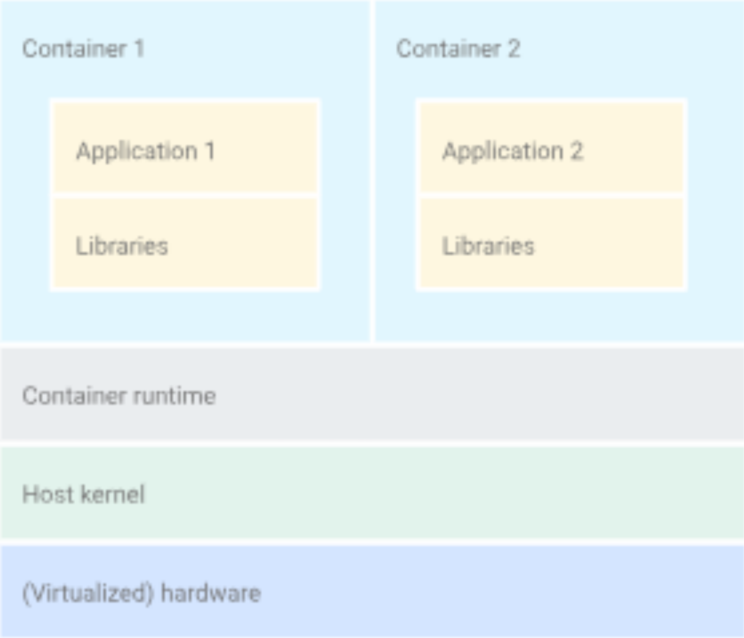
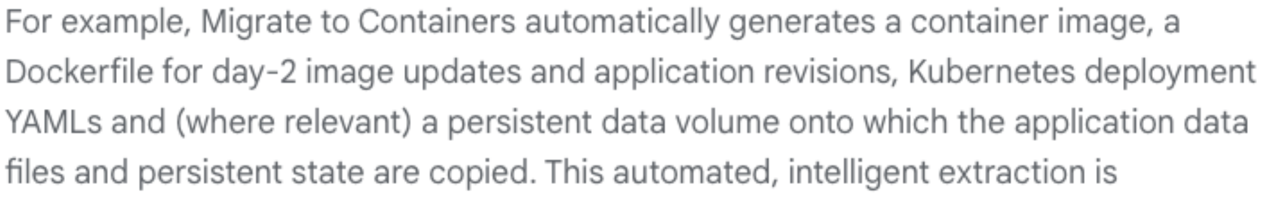
Claim 1	Accused Instrumentalities
	 Google containers platform Flexible deployment Container image App 2 Tomcat Services Persistent volume Networking Logging OS Kernel + drivers https://cloud.google.com/blog/products/application-modernization/shift-your-apps-to-container-based-workloads-on-the-command-line

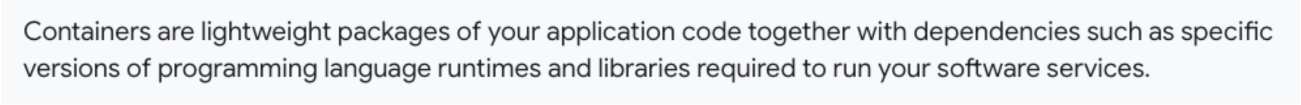
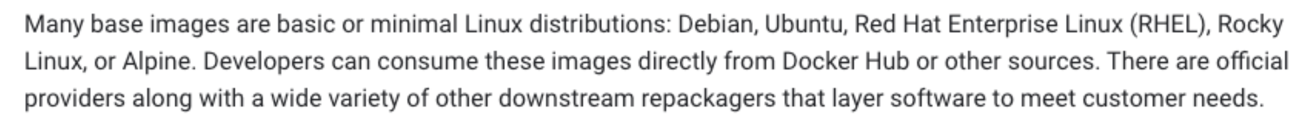
Claim 1	Accused Instrumentalities
	 The migration prerequisites are dependent on your specific migration environment. Confirm that your workloads' OS and source platform are compatible for migration by reviewing the prerequisites for your specific migration environment: https://cloud.google.com/migrate/containers/docs/setting-up-overview  Containers use specific features of the Linux kernel that "trick" individual applications into thinking they're in their own unique environment, even though multiple applications share the same host kernel. (If you're not familiar with the Linux kernel, it's a part of the operating system that communicates between processes--requests that do user tasks like opening a file, running a program-- and the hardware. It manages resources like memory and CPU to meet these requests). https://services.google.com/fh/files/misc/why_container_security_matters.pdf
[1c] c) a shared library having shared library critical system elements (SLCSEs) stored therein for use by the plurality of software applications in user mode and	Each Accused Instrumentality comprises a shared library having shared library critical system elements (SLCSEs) stored therein for use by the plurality of software applications in user mode. *See, e.g.*:

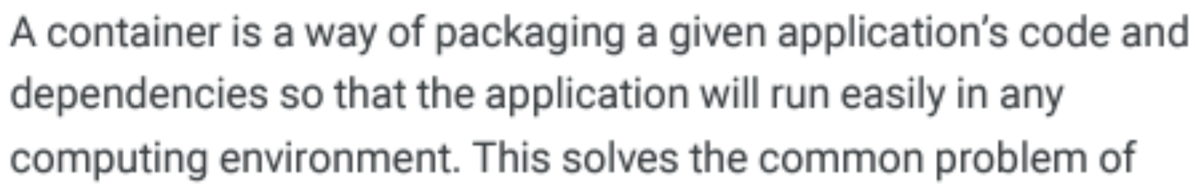
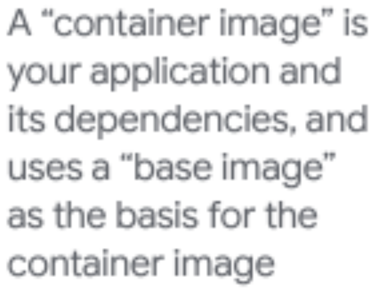
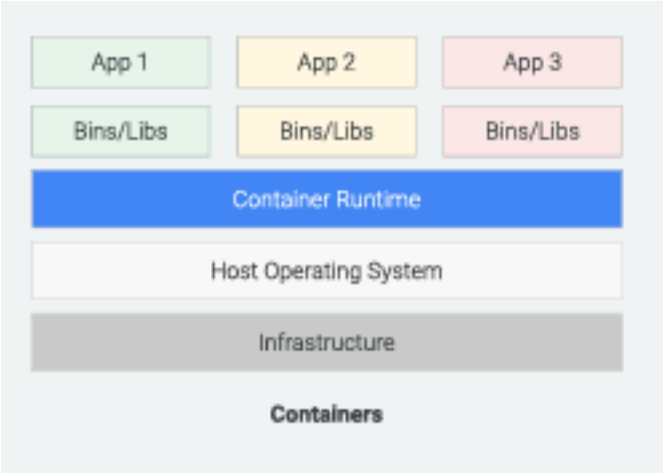
Claim 1	Accused Instrumentalities
	 A “container image” is your application and its dependencies, and uses a “base image” as the basis for the container image The container image specifies the container’s file system. For example, if you’re running a Node.js application, the container image would contain your app, Node.js, and other dependencies like Linux system libraries (except the kernel). A container image usually extends a base operating system image, or **base image**. This base image is the basis of your container, so you’ll want to ensure that it’s properly patched and free from known vulnerabilities. https://services.google.com/fh/files/misc/why_container_security_matters.pdf

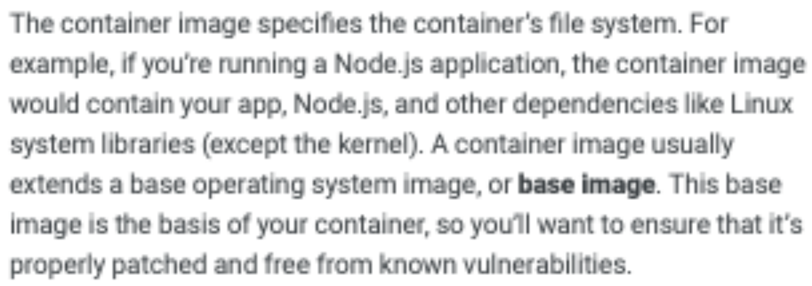
Claim 1	Accused Instrumentalities
	 A base image is the starting point for most container-based development workflows. Developers start with a base image and layer on top of it the necessary libraries, binaries, and configuration files used to run their application. Many base images are basic or minimal Linux distributions: Debian, Ubuntu, Red Hat Enterprise Linux (RHEL), Rocky Linux, or Alpine. Developers can consume these images directly from Docker Hub or other sources. There are official providers along with a wide variety of other downstream repackagers that layer software to meet customer needs. Google maintains base images for building its own applications. These images are built from the same source that Docker Hub uses. Therefore, they match the images you would get from Docker Hub. https://cloud.google.com/software-supply-chain-security/docs/base-images  The preconfigured base images provided by Cloud Workstations contain only a minimal environment with IDE, basic Linux terminal and language tools and a `sshd` server. To expedite the environment setup of specific development use cases, you can create custom container images that extend these base images to pre-install tools and dependencies and that run automation scripts. For custom container images, we recommend setting up a pipeline to automatically rebuild these images when the Cloud Workstations base image is updated, in addition to running a container scanning tool such as Artifact Analysis to inspect any additional dependencies you added. You're responsible for maintaining and updating custom packages and dependencies added to custom images. https://cloud.google.com/workstations/docs/customize-container-images  A container is a way of packaging a given application's code and dependencies so that the application will run easily in any computing environment. This solves the common problem of

Claim 1	Accused Instrumentalities
	 Containers solve the portability problem by isolating the application and its dependencies so they can be moved seamlessly between machines. A process running in a container lives isolated from the underlying environment. You control what it can see and what resources it can access. This helps you use resources more efficiently and not worry about the underlying infrastructure. One of the primary reasons to adopt containers is for your applications to be decoupled from the underlying environment and support higher resource utilization by "bin packing" multiple workloads onto each server. As such, the architecture of containers means that they're deployed with multiple containers sharing the same kernel.  The core components of the Linux kernel that are used for containers are **cgroups** — control groups, which define the resources like CPU and memory which are available to a given process — and **namespaces**, which are a way of separating processes by restricting what each process can see, so that system resources "appear" isolated to the process. https://services.google.com/fh/files/misc/why_container_security_matters.pdf

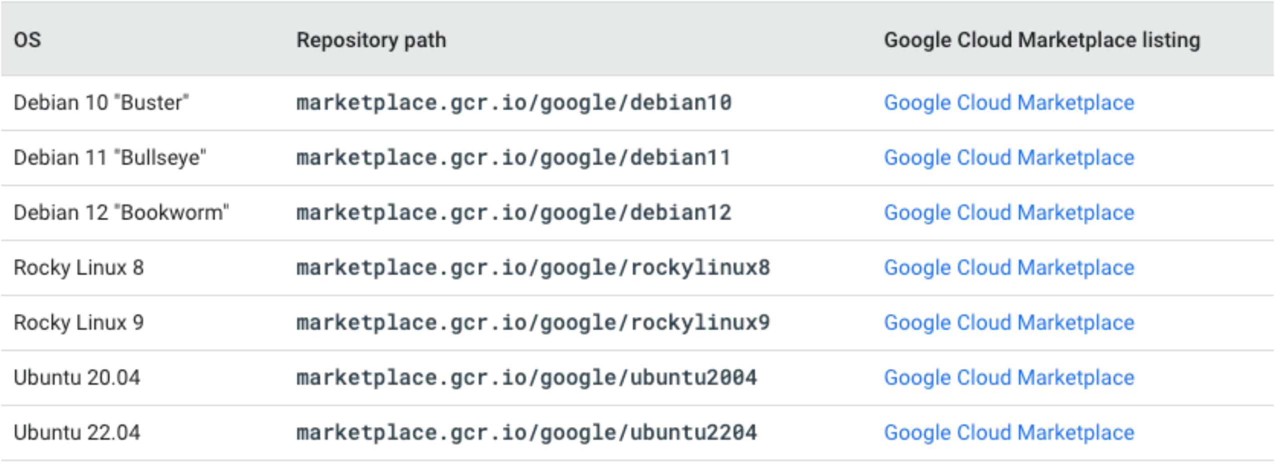
Claim 1	Accused Instrumentalities
	 Container 1 Application 1 Libraries Container 2 Application 2 Libraries Container runtime Host kernel (Virtualized) hardware https://cloud.google.com/architecture/best-practices-for-operating-containers  For example, Migrate to Containers automatically generates a container image, a Dockerfile for day-2 image updates and application revisions, Kubernetes deployment YAMLs and (where relevant) a persistent data volume onto which the application data files and persistent state are copied. This automated, intelligent extraction is https://cloud.google.com/blog/products/containers-kubernetes/how-migrate-for-anthos-improves-vm-to-container-migration

Claim 1	Accused Instrumentalities
	 Containers are lightweight packages of your application code together with dependencies such as specific versions of programming language runtimes and libraries required to run your software services. https://cloud.google.com/learn/what-are-containers
[1d] i) wherein some of the SLCSEs stored in the shared library are functional replicas of OSCSEs and are accessible to some of the plurality of software applications and when one of the SLCSEs is accessed by one or more of the plurality of software applications it forms a part of the one or more of the plurality of software applications,	In each Accused Instrumentality, some of the SLCSEs stored in the shared library are functional replicas of OSCSEs and are accessible to some of the plurality of software applications and when one of the SLCSEs is accessed by one or more of the plurality of software applications it forms a part of the one or more of the plurality of software applications. For example, a Docker base image serves as a self-contained unit that encompasses all the necessary components for an application to run, including the application code, runtime environment, system tools, and dependencies (i.e., SLCSEs). The images are based on existing Linux distributions, such as Debian and Ubuntu, including essential system elements (i.e., functional replicas of OSCSEs). Each container image is based on a specific base image, which contains the application code, and dependencies, including system libraries or shared library critical system elements (SLCSEs). When the container runs the image, it creates a runtime instance of that container image. *See, e.g.*:  Many base images are basic or minimal Linux distributions: Debian, Ubuntu, Red Hat Enterprise Linux (RHEL), Rocky Linux, or Alpine. Developers can consume these images directly from Docker Hub or other sources. There are official providers along with a wide variety of other downstream repackagers that layer software to meet customer needs. https://cloud.google.com/software-supply-chain-security/docs/base-images

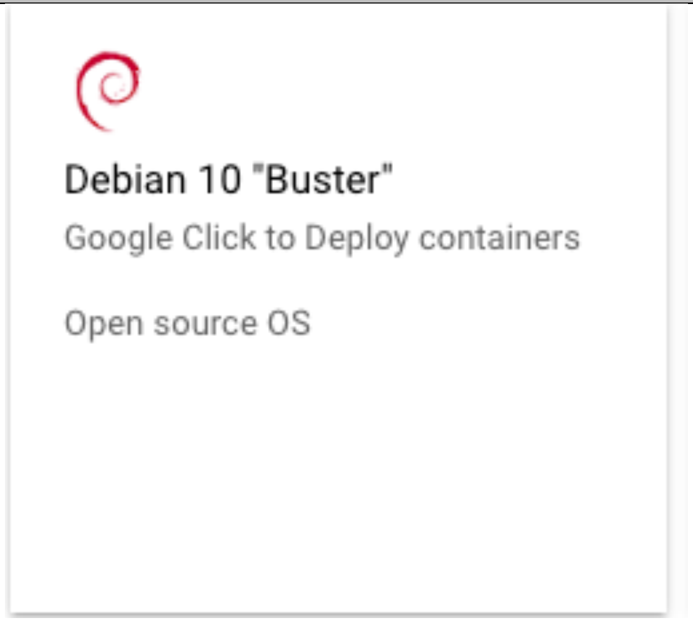
Claim 1	Accused Instrumentalities
	 A container is a way of packaging a given application's code and dependencies so that the application will run easily in any computing environment. This solves the common problem of  A "container image" is your application and its dependencies, and uses a "base image" as the basis for the container image  App 1 App 2 App 3 Bins/Libs Bins/Libs Bins/Libs Container Runtime Host Operating System Infrastructure Containers

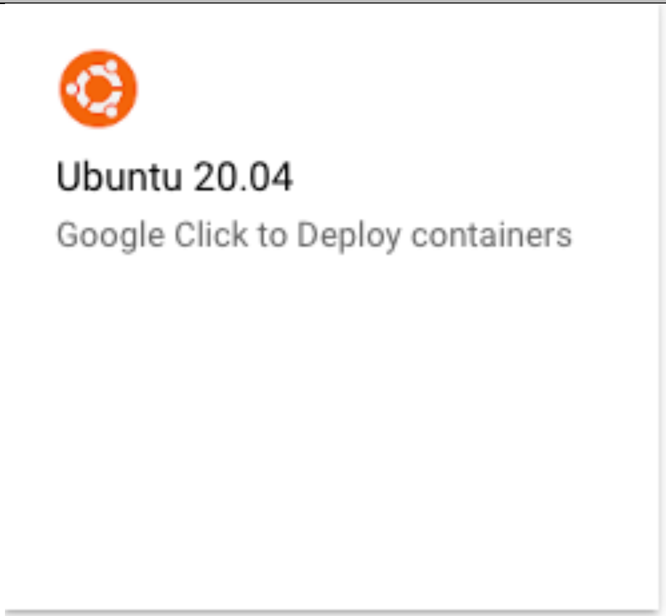
<table>
<tr><th>Claim 1</th><th>Accused Instrumentalities</th></tr>
<tr><td></td><td>

The container image specifies the container's file system. For example, if you're running a Node.js application, the container image would contain your app, Node.js, and other dependencies like Linux system libraries (except the kernel). A container image usually extends a base operating system image, or <b>base image</b>. This base image is the basis of your container, so you'll want to ensure that it's properly patched and free from known vulnerabilities.

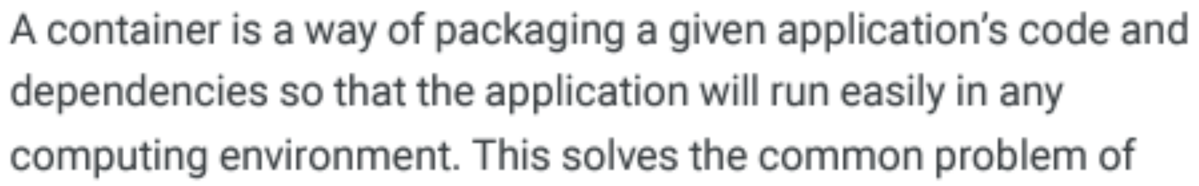
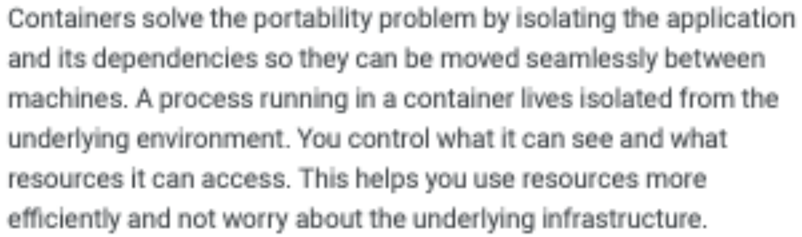
https://services.google.com/fh/files/misc/why_container_security_matters.pdf

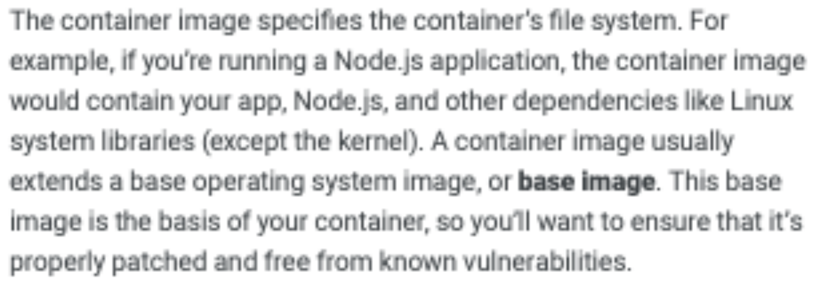
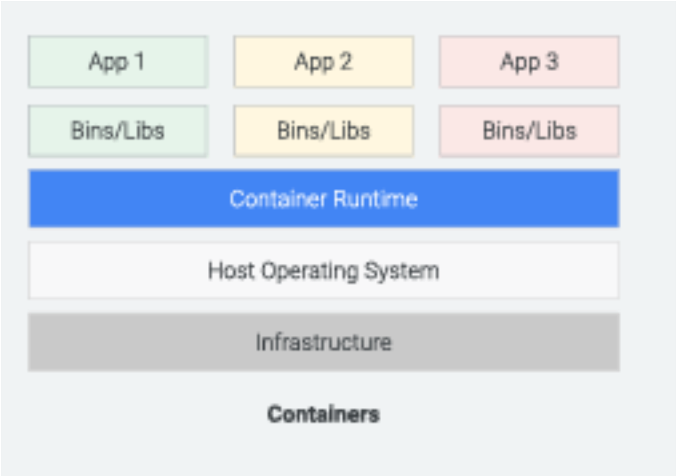

<table>
<tr><th>OS</th><th>Repository path</th><th>Google Cloud Marketplace listing</th></tr>
<tr><td>Debian 10 "Buster"</td><td>marketplace.gcr.io/google/debian10</td><td>Google Cloud Marketplace</td></tr>
<tr><td>Debian 11 "Bullseye"</td><td>marketplace.gcr.io/google/debian11</td><td>Google Cloud Marketplace</td></tr>
<tr><td>Debian 12 "Bookworm"</td><td>marketplace.gcr.io/google/debian12</td><td>Google Cloud Marketplace</td></tr>
<tr><td>Rocky Linux 8</td><td>marketplace.gcr.io/google/rockylinux8</td><td>Google Cloud Marketplace</td></tr>
<tr><td>Rocky Linux 9</td><td>marketplace.gcr.io/google/rockylinux9</td><td>Google Cloud Marketplace</td></tr>
<tr><td>Ubuntu 20.04</td><td>marketplace.gcr.io/google/ubuntu2004</td><td>Google Cloud Marketplace</td></tr>
<tr><td>Ubuntu 22.04</td><td>marketplace.gcr.io/google/ubuntu2204</td><td>Google Cloud Marketplace</td></tr>
</table>

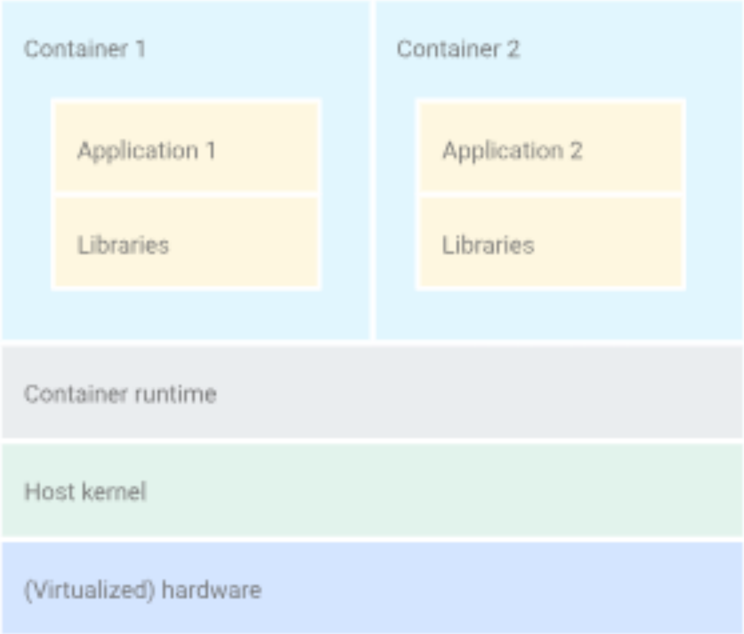
https://cloud.google.com/software-supply-chain-security/docs/base-images
</td></tr>
</table>

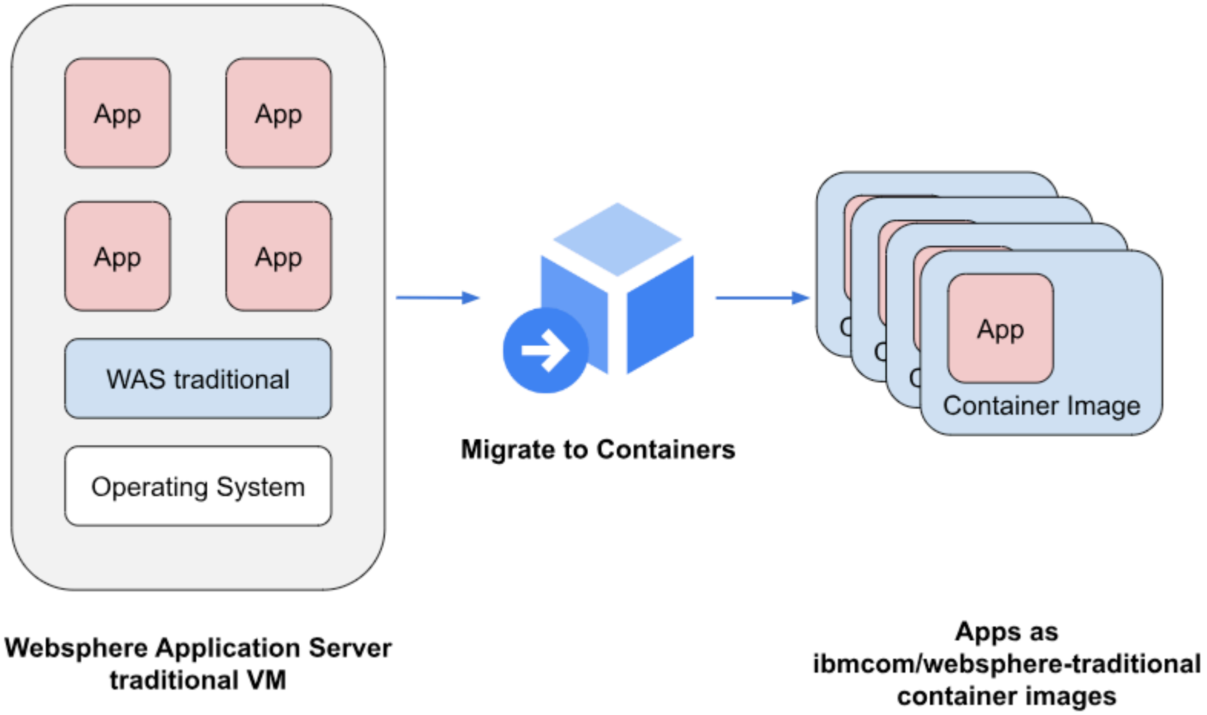
Claim 1	Accused Instrumentalities
	 Debian 10 "Buster" Google Click to Deploy containers Open source OS

Claim 1	Accused Instrumentalities
	 Ubuntu 20.04 Google Click to Deploy containers https://console.cloud.google.com/marketplace/browse?filter=solution-type:container
[1e] ii) wherein an instance of a SLCSE provided to at least a first of the plurality of software applications from the shared library is run in a context of said at least first of the plurality of software applications without being shared with other of the plurality of software applications and where at least a second of the plurality of software applications running	In each Accused Instrumentality, an instance of a SLCSE provided to at least a first of the plurality of software applications from the shared library is run in a context of said at least first of the plurality of software applications without being shared with other of the plurality of software applications and where at least a second of the plurality of software applications running under the operating system have use of a unique instance of a corresponding critical system element for performing same function. When a Docker image is used to create a container, it creates a separate and isolated instance of a runtime environment which is independent of other containers running on the same host. Each container has its own instance of base images and its own data. The containers run in isolation, ensuring that the SLCSEs stored in the shared library are accessible to the software applications running in their respective containers. The docker image includes essential system files, libraries, and dependencies required to run the software application within the container. The Docker

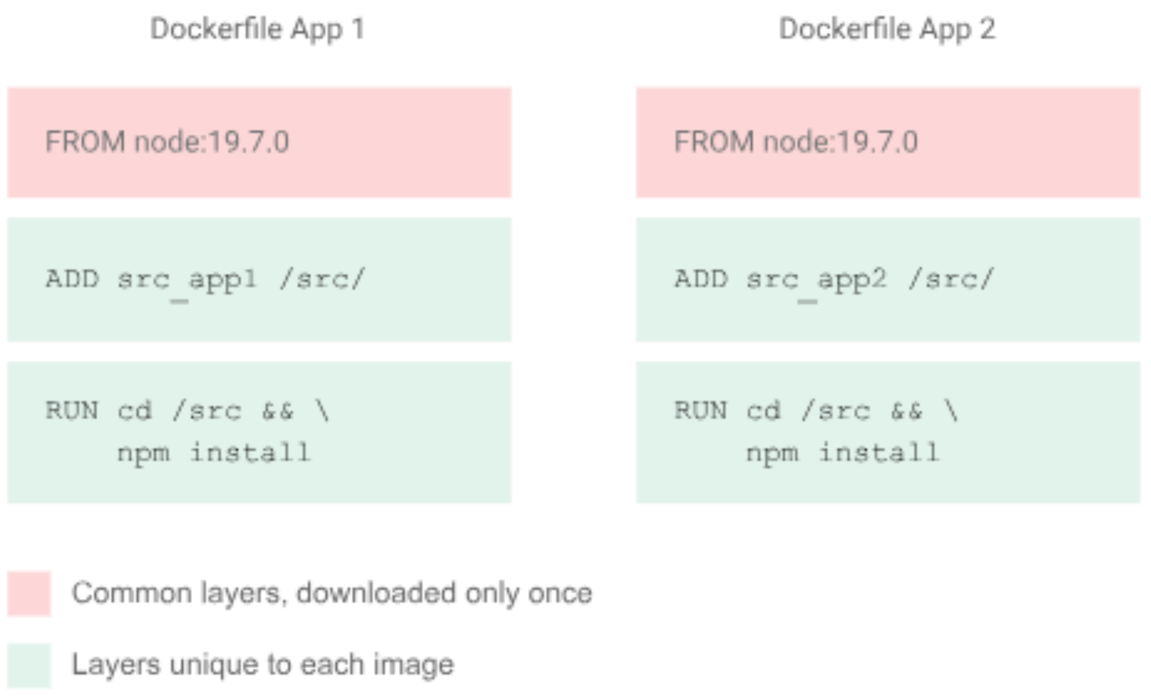
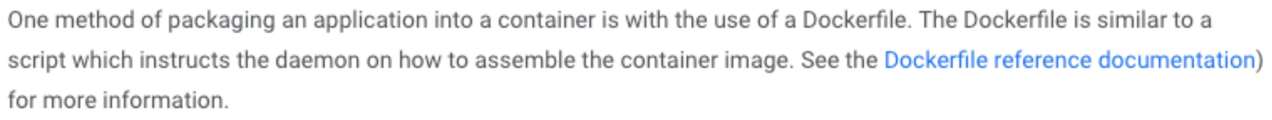
Claim 1	Accused Instrumentalities
under the operating system have use of a unique instance of a corresponding critical system element for performing same function, and	containers can share common dependencies and components using layered images. This means that multiple containers utilize the same base image to create an instance. When an instance of SLCSE is provided from the base image (i.e., from the shared library) to an individual container including application software, it operates in isolation and runs its own instance of the software application without sharing resources or critical system elements with other containers. This ensures that each container has its own isolated context. Docker containers can share common dependencies and components using layered images. This means that multiple containers can utilize the same base image. Therefore, each container, containing the application software running under the operating system, utilizes a unique instance of the corresponding critical system element to execute the respective application software for performing a same or a different function. *See, e.g.*:  A container is a way of packaging a given application's code and dependencies so that the application will run easily in any computing environment. This solves the common problem of  Containers solve the portability problem by isolating the application and its dependencies so they can be moved seamlessly between machines. A process running in a container lives isolated from the underlying environment. You control what it can see and what resources it can access. This helps you use resources more efficiently and not worry about the underlying infrastructure.

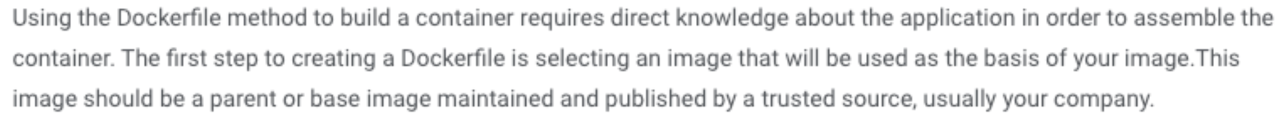
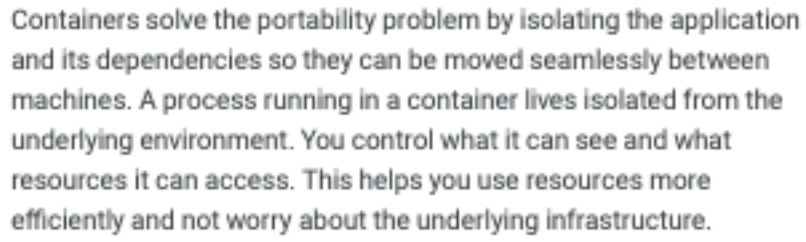
Claim 1	Accused Instrumentalities
	 The container image specifies the container's file system. For example, if you're running a Node.js application, the container image would contain your app, Node.js, and other dependencies like Linux system libraries (except the kernel). A container image usually extends a base operating system image, or **base image**. This base image is the basis of your container, so you'll want to ensure that it's properly patched and free from known vulnerabilities.  App 1 App 2 App 3 Bins/Libs Bins/Libs Bins/Libs Container Runtime Host Operating System Infrastructure Containers https://services.google.com/fh/files/misc/why_container_security_matters.pdf

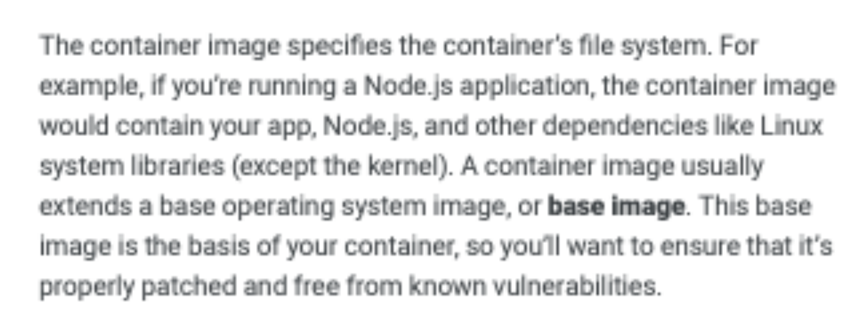
Claim 1	Accused Instrumentalities
	 Container 1 Application 1 Libraries Container 2 Application 2 Libraries Container runtime Host kernel (Virtualized) hardware https://cloud.google.com/architecture/best-practices-for-operating-containers

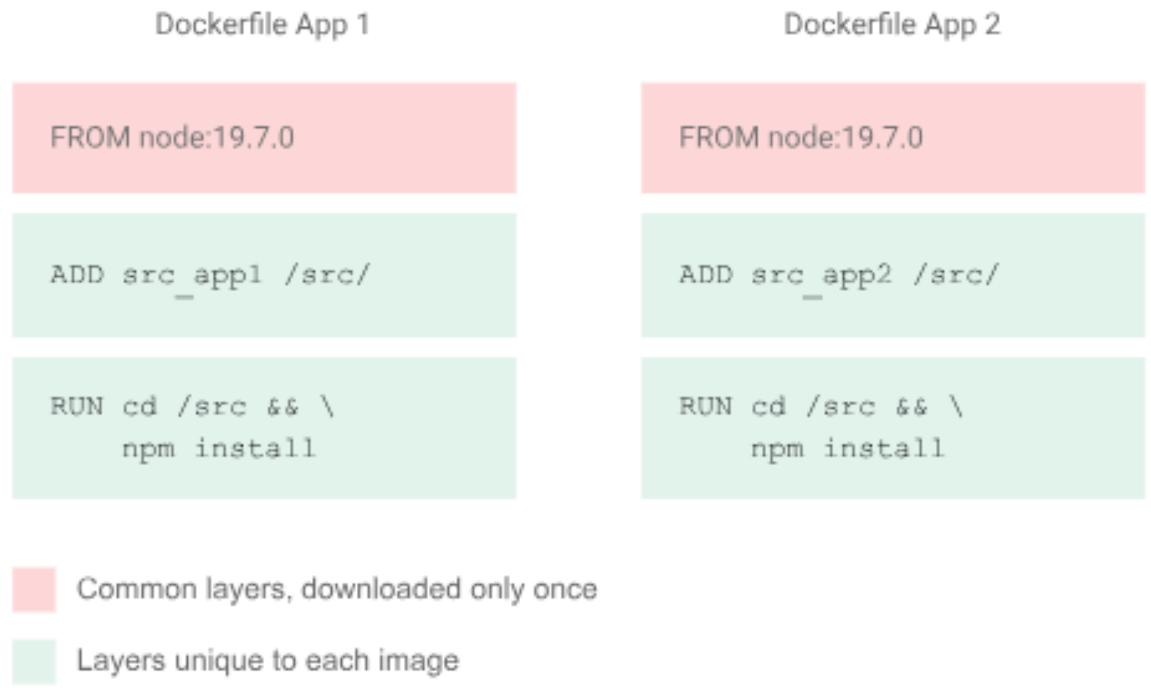
Claim 1	Accused Instrumentalities
	 App App App App WAS traditional Operating System Websphere Application Server traditional VM Migrate to Containers App Container Image Apps as ibmcom/websphere-traditional container images https://cloud.google.com/migrate/containers/docs/migrating-overview

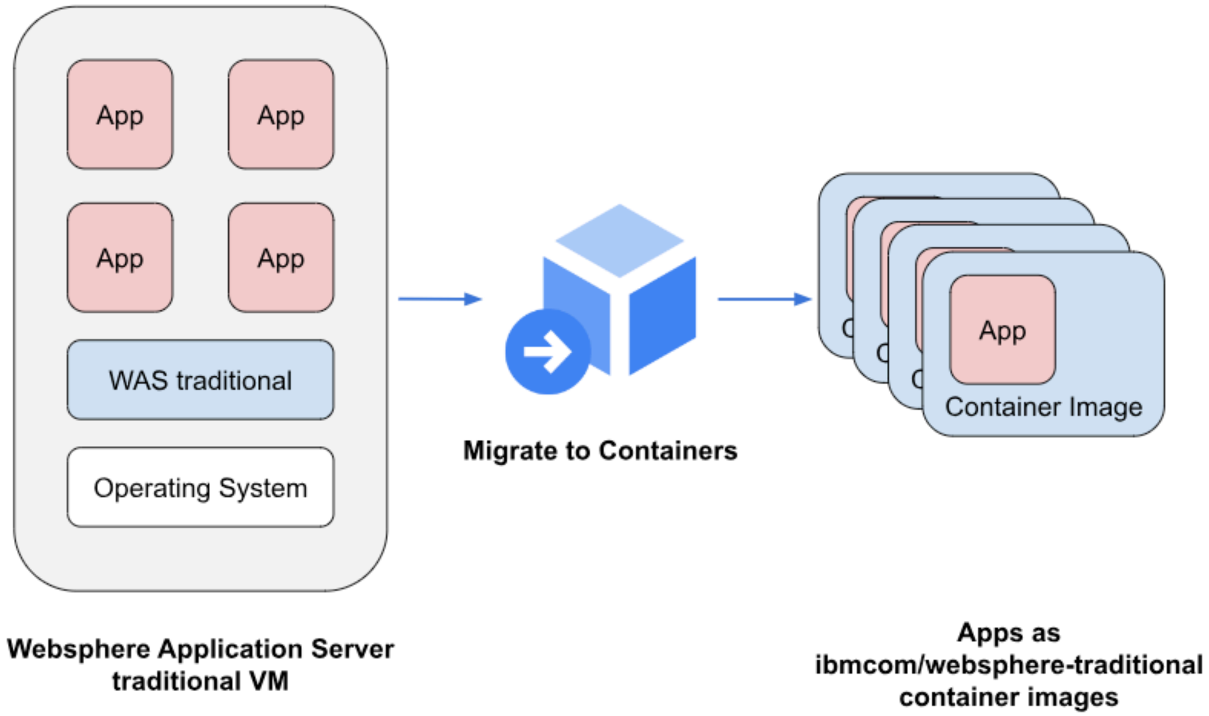
Claim 1	Accused Instrumentalities

	
	Dockerfile App 1
	FROM node:19.7.0
	`ADD src_app1 /src/`
	`RUN cd /src && \`
	`    npm install`
	Dockerfile App 2
	FROM node:19.7.0
	`ADD src_app2 /src/`
	`RUN cd /src && \`
	`    npm install`
	🟥 Common layers, downloaded only once
	🟩 Layers unique to each image
	https://cloud.google.com/architecture/best-practices-for-building-containers
	
	One method of packaging an application into a container is with the use of a Dockerfile. The Dockerfile is similar to a script which instructs the daemon on how to assemble the container image. See the Dockerfile reference documentation) for more information.

Claim 1	Accused Instrumentalities
	 Using the Dockerfile method to build a container requires direct knowledge about the application in order to assemble the container. The first step to creating a Dockerfile is selecting an image that will be used as the basis of your image.This image should be a parent or base image maintained and published by a trusted source, usually your company. https://codelabs.developers.google.com/developing-containers-with-dockerfiles#2
[1f] iii) wherein a SLCSE related to a predetermined function is provided to the first of the plurality of software applications for running a first instance of the SLCSE, and wherein a SLCSE for performing a same function is provided to the second of the plurality of software applications for running a second instance of the SLCSE simultaneously.	In each Accused Instrumentality, a SLCSE related to a predetermined function is provided to the first of the plurality of software applications for running a first instance of the SLCSE, and wherein a SLCSE for performing a same function is provided to the second of the plurality of software applications for running a second instance of the SLCSE simultaneously. For example, In Docker, each container operates independently, and a Docker base image includes essential system files, libraries, and dependencies (i.e., SLCSEs) required to run the software application within the container. Based on information and belief, each element, such as system files, libraries, and dependencies (i.e., SLCSE) is associated with an execution of a predetermined function related to the application. When a Docker image is used to create a container in ECS, an instance of the SLCSE is provided to a software application. Therefore, different instances of the SLCSE are provided to different applications for performing either a same or a different function, simultaneously. *See, e.g.*:  Containers solve the portability problem by isolating the application and its dependencies so they can be moved seamlessly between machines. A process running in a container lives isolated from the underlying environment. You control what it can see and what resources it can access. This helps you use resources more efficiently and not worry about the underlying infrastructure.

Claim 1	Accused Instrumentalities
	 The container image specifies the container's file system. For example, if you're running a Node.js application, the container image would contain your app, Node.js, and other dependencies like Linux system libraries (except the kernel). A container image usually extends a base operating system image, or **base image**. This base image is the basis of your container, so you'll want to ensure that it's properly patched and free from known vulnerabilities. https://services.google.com/fh/files/misc/why_container_security_matters.pdf

Claim 1	Accused Instrumentalities
	 FROM node:19.7.0 ADD src_app1 /src/ RUN cd /src && \ npm install FROM node:19.7.0 ADD src_app2 /src/ RUN cd /src && \ npm install 🟥 Common layers, downloaded only once 🟩 Layers unique to each image https://cloud.google.com/architecture/best-practices-for-building-containers

Claim 1	Accused Instrumentalities
	 App App App App WAS traditional Operating System Websphere Application Server traditional VM Migrate to Containers App Container Image Apps as ibmcom/websphere-traditional container images https://cloud.google.com/migrate/containers/docs/migrating-overview